Terms used **escape analysis** Found **75,594** of **178,880**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Pointer and escape analysis for multithreaded programs
Alexandru Salcianu, Martin Rinard
June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming PPoPP '01**, Volume 36 Issue 7
Publisher: ACM Press
Full text available: pdf(318.11 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new combined pointer and escape analysis for multithreaded programs. The algorithm uses a new abstraction called parallel interaction graphs to analyze the interactions between threads and extract precise points-to, escape, and action ordering information for objects accessed by multiple threads. The analysis is compositional, analyzing each method or thread once to extract a parameterized analysis result that can be specialized for use in any context. ...

2 Incrementalized pointer and escape analysis
Frédéric Vivien, Martin Rinard
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation PLDI '01**, Volume 36 Issue 5
Publisher: ACM Press
Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a new pointer and escape analysis. Instead of analyzing the whole program, the algorithm incrementally analyzes only those parts of the program that may deliver useful results. An analysis policy monitors the analysis results to direct the incremental investment of analysis resources to those parts of the program that offer the highest expected optimization return.

Our experimental results show that almost all of the objects are allocated at a small number of allocation sit ...

3 Compositional pointer and escape analysis for Java programs
John Whaley, Martin Rinard
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(2.42 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a combined pointer and escape analysis algorithm for Java programs. The algorithm is based on the abstraction of points-to escape graphs, which characterize how local variables and fields in objects refer to other objects. Each points-to escape graph also contains escape information, which characterizes how objects allocated in one region of the program can escape to be accessed by another region. The algorithm is designed to analyze arbitrary regions of complete or inco ...

4 Escape analysis for object-oriented languages: application to Java

Bruno Blanchet
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

Escape analysis [27, 14, 5] is a static analysis that determines whether the lifetime of data exceeds its static scope.The main originality of our escape analysis is that it determines precisely the effect of assignments, which is necessary to apply it to object oriented languages with promising results, whereas previous work [27, 14, 5] applied it to functional languages and were very imprecise on assignments. Our implementation analyses the full Java™ Language. ...

5 Escape analysis for Java

Jong-Deok Choi, Manish Gupta, Mauricio Serrano, Vugranam C. Sreedhar, Sam Midkiff
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(1.85 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a simple and efficient data flow algorithm for escape analysis of objects in Java programs to determine (i) if an object can be allocated on the stack; (ii) if an object is accessed only by a single thread during its lifetime, so that synchronization operations on that object can be removed. We introduce a new program abstraction for escape analysis, the connection graph, that is used to establish reachability relationships between objects and object ref ...

6 Escape analysis on lists

Young Gil Park, Benjamin Goldberg
July 1992 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation PLDI '92**, Volume 27 Issue 7

Publisher: ACM Press

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

Higher order functional programs constantly allocate objects dynamically. These objects are typically cons cells, closures, and records and are generally allocated in the heap and reclaimed later by some garbage collection process. This paper describes a compile time analysis, called escape analysis, for determining the lifetime of dynamically created objects in higher order functional programs, and describes optimizations that can be performed, based on the analysis, to improve storage all ...

7 Reference escape analysis: optimizing reference counting based on the lifetime of references

Young Gil Park, Benjamin Goldberg

May 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation PEPM '91**, Volume 26 Issue 9

Publisher: ACM Press

Full text available: pdf(1.06 MB) Additional Information: full citation, references, citings, index terms

8 Stack allocation and synchronization optimizations for Java using escape analysis

Jong-Deok Choi, Manish Gupta, Mauricio J. Serrano, Vugranam C. Sreedhar, Samuel P. Midkiff

November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6

Publisher: ACM Press

Full text available: pdf(632.85 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an *escape analysis* framework for Java to determine (1) if an object is not reachable after its method of creation returns, allowing the object to be allocated on the stack, and (2) if an object is reachable only from a single thread during its lifetime, allowing unnecessary synchronization operations on that object to be removed. We introduce a new program abstraction for escape analysis, the *connection graph*, that is used to establish reachability relationshi ...

Keywords: Connection graphs, escape analysis, points-to graph

9 Escape analysis for Java™: Theory and practice

Bruno Blanchet

November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6

Publisher: ACM Press

Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its static scope.This paper first presents the design and correctness proof of an escape analysis for Java™. This analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment form. So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imperative language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

10 Dynamic compilation techniques: Escape analysis in the context of dynamic compilation and deoptimization

Thomas Kotzmann, Hanspeter Mössenböck

June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments**

Publisher: ACM Press

Full text available: Additional Information:

pdf(203.40 KB) full citation, abstract, references, index terms

In object-oriented programming languages, an object is said to *escape* the method or thread in which it was created if it can also be accessed by other methods or threads. Knowing which objects do not escape allows a compiler to perform aggressive optimizations.This paper presents a new intraprocedural and interprocedural algorithm for escape analysis in the context of dynamic compilation where the compiler has to cope with dynamic class loading and deoptimization. It was implemented for S ...

Keywords: Java, deoptimization, escape analysis, just-in-time compilation, optimization, scalar replacement, stack allocation, synchronization removal

11 Escape analysis: correctness proof, implementation and experimental results

Bruno Blanchet

January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(1.68 MB) Additional Information: full citation, references, citings, index terms

12 Field analysis: getting useful and low-cost interprocedural information

Sanjay Ghemawat, Keith H. Randall, Daniel J. Scales

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(686.96 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new limited form of interprocedural analysis called field analysis that can be used by a compiler to reduce the costs of modern language features such as object-oriented programming, automatic memory management, and run-time checks required for type safety. Unlike many previous interprocedural analyses, our analysis is cheap, and does not require access to the entire program. Field analysis exploits the declared access restrictions placed on fields in a modul ...

13 Access control analysis: Data-centric security: role analysis and role typestates

Vugranam C. Sreedhar

June 2006 **Proceedings of the eleventh ACM symposium on Access control models and technologies SACMAT '06**

Publisher: ACM Press

Full text available: pdf(270.98 KB) Additional Information: full citation, abstract, references, index terms

In J2EE and .NET roles are assigned to methods using external configuration files, called the deployment descriptors. Assigning roles to methods, although conceptually simple, in practice it is quite complicated. For instance, in order for a deployer to assign a role *r* to a method *m*, the deployer must understand the set of roles *R* that are assigned to each method *n* that can be invoked directly or indirectly from *m*, and that *r* has to be "consistently" ass ...

Keywords: RBAC, role analysis, role escape analysis, role typestates

14 Interprocedural compatibility analysis for static object preallocation

Ovidiu Gheorghioiu, Alexandru Salcianu, Martin Rinard

January 2003 **ACM SIGPLAN Notices , Proceedings of the 30th ACM SIGPLAN-SIGACT**

symposium on Principles of programming languages POPL '03, Volume 38 Issue 1
Publisher: ACM Press

Full text available: pdf(277.65 KB) Additional Information: full citation, abstract, references, citings, index terms

We present an interprocedural and compositional algorithm for finding pairs of *compatible* allocation sites, which have the property that no object allocated at one site is live at the same time as any object allocated at the other site. If an allocation site is compatible with itself, it is said to be *unitary*: at most one object allocated at that site is live at any given point in the, execution of the program. We use the results of the analysis to statically preallocate memory spa ...

Keywords: interprocedural analysis, memory preallocation, static analysis

15 Cloning-based context-sensitive pointer alias analysis using binary decision diagrams
John Whaley, Monica S. Lam
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation PLDI '04**, Volume 39 Issue 6
Publisher: ACM Press

Full text available: pdf(277.87 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the first scalable context-sensitive, inclusion-based pointer alias analysis for Java programs. Our approach to context sensitivity is to create a clone of a method for every context of interest, and run a *context-insensitive* algorithm over the expanded call graph to get *context-sensitive* results. For precision, we generate a clone for every acyclic path through a program's call graph, treating methods in a strongly connected component as a single node. Normally ...

Keywords: Datalog, Java, binary decision diagrams, cloning, context-sensitive, inclusion-based, logic programming, pointer analysis, program analysis, scalable

16 On the complexity of set-based analysis
Nevin Heintze, David McAllester
August 1997 **ACM SIGPLAN Notices , Proceedings of the second ACM SIGPLAN international conference on Functional programming ICFP '97**, Volume 32 Issue 8
Publisher: ACM Press

Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, index terms

We define a general notion of set-based analysis --- any language whose operational semantics is defined by environment evaluation has a well defined set-based abstraction. This general definition covers both Aiken and Wimmers' type system and Heintze' set-based analysis. Aiken and Wimmers give a nondeterministic exponential time algorithm for their analysis. Heintze gives an $O(n^3)$ procedure. We show that this discrepancy is due to the complexity of the case statements a ...

17 Static conflict analysis for multi-threaded object-oriented programs
Christoph von Praun, Thomas R. Gross
May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI '03**, Volume 38 Issue 5
Publisher: ACM Press

Full text available: pdf(674.11 KB) Additional Information: full citation, abstract, references, citings, index

terms

A compiler for multi-threaded object-oriented programs needs information about the sharing of objects for a variety of reasons: to implement optimizations, to issue warnings, to add instrumentation to detect access violations that occur at runtime. An Object Use Graph (OUG) statically captures accesses from different threads to objects. An OUG extends the Heap Shape Graph (HSG), which is a compile-time abstraction for runtime objects (nodes) and their reference relations (edges). An OUG specifie ...

Keywords: heap shape graph, object use graph, program analysis, race detection, representations for concurrent programs

18 High-level data flow analysis

Barry K. Rosen
October 1977 **Communications of the ACM**, Volume 20 Issue 10
Publisher: ACM Press
Full text available: pdf(1.18 MB) Additional Information: full citation, abstract, references, citings

In contrast to the predominant use of low-level intermediate text, high-level data flow analysis deals with programs essentially at source level and exploits the control flow information implicit in the parse tree. The need for high-level flow analysis arises from several aspects of recent work on advanced methods of program certification and optimization. This paper proposes a simple general method of high-level data flow analysis that allows free use of escape and jump statements, avoids ...

Keywords: control flow graph, data flow analysis, escapes, exits, goto statements, high-level language, jumps, structured programming

19 Resource usage analysis

Atsushi Igarashi, Naoki Kobayashi
January 2002 **ACM SIGPLAN Notices , Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '02**, Volume 37 Issue 1
Publisher: ACM Press
Full text available: pdf(225.18 KB) Additional Information: full citation, abstract, references, citings

It is an important criterion of program correctness that a program accesses resources in a valid manner. For example, a memory region that has been allocated should be eventually deallocated, and after the deallocation, the region should no longer be accessed. A file that has been opened should be eventually closed. So far, most of the methods to analyze this kind of property have been proposed in rather specific contexts (like studies of memory management and verification of usage of lock primi ...

20 A classification system and analysis for aspect-oriented programs

Martin Rinard, Alexandru Salcianu, Suhabe Bugrara
October 2004 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering SIGSOFT '04/FSE-12**, Volume 29 Issue 6
Publisher: ACM Press
Full text available: pdf(247.39 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new classification system for aspect-oriented programs. This system characterizes the interactions between aspects and methods and identifies classes of interactions that enable modular reasoning about the crosscut program. We argue that this system can help developers structure their understanding of aspect-oriented

programs and promotes their ability to reason productively about the consequences of crosscutting a program with a given aspect.

We have designed and implemen ...

Keywords: aspect oriented programming, program analysis

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next